Implementation Guide

WSP ACCESS

WSP ACCESS System
Extended Validation String
Implementation Guide

This document contains information, specifications and diagrams of a highly proprietary and confidential nature. This information is intended only for use by the organization, to which it was distributed directly by Datamaxx Group, Inc. Under no circumstances is there to be any duplication, distribution or other release of the information contained in this document to any other organization or person, by any means, without written authorization from Datamaxx Group, Inc.

Datamaxx®
APPLIED TECHNOLOGIES

Published by:

Datamaxx Applied Technologies, Inc.
2001 Drayton Drive
Tallahassee, FL 32311-7854
(850) 558-8000
www.Datamaxx.com

Revision History:

Version	Date	Notes
Version 1.0	07/09/2018	Original Document
Version 1.1	08/13/2018	Clarified Hash algorithm and use of encryption. Added pseudo code to show the creation of the Extended Validation String. Minor typos and edits

Table of Contents

1.0 INTRODUCTION

The purpose of this document is to provide personnel implementing the secure, encrypted communications access to the Washington State Patrol ACCESS system information as to the programming required to create the “Extended Validation String”.

The “Extended Validation String” is a methodology whereby a remote system can identify itself to the WSP ACCESS system independent of the network interfaces, or defined hard coded TCP/IP addresses. The FBI CJIS Security policy requires that all end points be positively identified to a system. When the network has intermediate functions such as a proxy server, or Network Address translation, or a threat management gateway it may not be possible to identify the end point by IP address. Thus a different method is required.

In the current implementation, the remote system is identified by a 1 to 5 character data string (e.g. “SEAXX”) that identifies a remote sever. This is sent in clear text over a hardware encrypted communications link. In the new implementation, the hardware will not perform encryption, and that will be handled by the communication protocol. In conjunction with the encryption, the identifier must be encrypted and other security aspects implemented.

This document describes the process for masking the actual data of the identifier and its encryption, thus creating the “Extended Validation String”.

2.0 COMMUNCIATIONS PROTOCOL

The communication protocol is the industry standard Datamaxx DMPP-2020 protocol, using the FIPS 140-2 encryption with a 256 bit key. ***There are no changes required to the protocol implementation itself.***

The encryption is a prerequisite and must be available before implementing the Extended Validation String as described in this document.

In the current implementation, the identifier (known as the Validation String "Val String") is sent in clear text upon establishment of a connection, independent of the data encryption setting that is used for the data messages. This is not secure in the future environments and is replaced with the Extended Validation string.

With the Extended Validation String, the identifier that is created (as noted below) will be sent encrypted, using the same encryption as applied to the actual data messages. This is a major distinction between the current implementation and the use of the Extended Validation String.

3.0 CREATING THE EXTENDED VALIDATION STRING

The Extended Validation String is created from the identifier supplied by the WSP administrators. It will not be the name of the remote system (as in the current implementation), but a unique data string generated for each system.

Note that this identifier is case sensitive.

To create the Extended Validation string, the following steps are performed programmatically.

1. Get the UTC time of the system in the format “yyyymmddhhmmss”.
2. Take the identifier provided by the WSP administrators and prepend it with the character string “OMNIXX”.
3. Create a one-way hash of the identifier as calculated in Step 2 and the time stamp. The hash algorithm is the industry standard HMAC-SH256.
4. Convert the result of the one way hash to Base 64 encoding. The time stamp will be the “secret key” for the hash function.
5. Create a final identifier by prepending the result of the Base64 encoding with the time stamp and the character “|” (vertical bar).

The result of step 5 is then sent upon initial connection, using the standard encryption as would be used for all data messages. The DMPP header function must be "01" (data message with no acknowledgement, final block).

The encryption is critical and is a difference from the previous implementation of the Validation String.

4.0 PROGAMMING EXAMPLE:

The following is an example of the creation of the Extended Validation string.

For the purposes of this example, the identifier as provided by the WSP administrative staff is “IAMABIGBADLONGIDENTIFIER”.

The pseudo code shows each step in the process.

The examples below show the data transformations step by step, with both the character and hexadecimal data shown in hex dump format.

```
UTC Timestamp:
32 30 31 38 30 37 32 33 30 30 32 33 34 33         20180723002343

Pre-pended Identifier:
4f 4d 4e 49 58 58 49 41 4d 41 42 49 47 42 41 44   OMNIXXIAMABIGBAD
4c 4f 4e 47 49 44 45 4e 54 49 46 49 45 52         LONGIDENTIFIER

PsuedoCode:
Prepended_Identifier = "OMNIXX" + Identifier;

Hash Result:
03 26 41 a2 54 ae 19 2e fc 95 00 b1 61 d4 ed e3   .&A.T.......a...
a8 6e b0 dd 0c 5e 6a 9d fc c3 eb f9 54 29 d6 43   .n...^j.....T).C

PsuedoCode:
Hashed_Result = HMAC_SH256(Prepended_Identifier, Timestamp);

Base 64 encoded Hashed Data:
41 79 5a 42 6f 6c 53 75 47 53 37 38 6c 51 43 78   AyZBolSuGS78lQCx
59 64 54 74 34 36 68 75 73 4e 30 4d 58 6d 71 64   YdTt46husN0MXmqd
2f 4d 50 72 2b 56 51 70 31 6b 4d 3d               /MPr+VQp1kM=

PsuedoCode:
Encoded_Data = ConvertBase64(Hashed_Result);

Final Extended Val String For Transmission:
32 30 31 38 30 37 32 33 30 30 32 33 34 33 7c 41   20180723002343|A
79 5a 42 6f 6c 53 75 47 53 37 38 6c 51 43 78 59   yZBolSuGS78lQCxY
64 54 74 34 36 68 75 73 4e 30 4d 58 6d 71 64 2f   dTt46husN0MXmqd/
4d 50 72 2b 56 51 70 31 6b 4d 3d

PsuedoCode:
Final_Validation String = Timestamp + "|" + Encoded_Data
```